

CLAIMS

What is claimed is:

1. A method of operating a computer to provide a floor planning tool to an integrated circuit designer, comprising the steps:
 - A) storing the data structures of a logical netlist;
 - B) displaying on one portion of a computer display a representation of the instances defined by said logical netlist;
 - C) providing one or more tools a user can invoke to create and locate pblocks on a floorplan of an integrated circuit being designed, said floor plan comprising one or more pblocks which may be nested to establish a physical hierarchy, and responding to invocation of one or more of said tools by creating one or more said pblock(s), each pblock represented by a data object having a predetermined structure, said floorplan being displayed on the same computer display as said representation of said instances defined by said logical netlist;
 - D) providing one or more tools a user can invoke to assign instances from said displayed representation of instances defined by said logical netlist into pblocks in said displayed hierarchy of pblocks;
 - E) responding to such assignment operations by changing the data in said data objects representing said pblocks to reflect which instances are assigned to each pblock; and
 - F) further responding to such assignment operations by determining the original connectivity between instances defined in said logical netlist and automatically changing data in predetermined data objects of said physical hierarchy so as to recreate said original connectivity by creating new nets and new pins as necessary which recreate said original connectivity.
2. The method of claim 1 wherein said assignment operations comprise selecting and dragging instances from the displayed logical hierarchy into displayed pblocks and dropping said instance in said pblock to assign said instance thereto by releasing a pointing tool button after the instance has been dragged to said pblock, and further comprising the step of exporting floorplan directives regarding physical

placements of all instances from said logical netlist based upon the assignment of said instances to pblocks in said physical hierarchy.

3. The process of claim 1 wherein step E comprises the steps:

marking instToAppend entries designating instances from said logical netlist which have been moved to a pblock with a pointer to the pblock to each instance has been moved, said pointer being in an array

m_instanceAssignments (hereafter referred to as the array) which defines which instances are assigned to each pblock;

marking all child instances in said logical hierarchy of each said instToAppend in said array to point to the same pblock to which said instToAppend was assigned;

recurring up the logical hierarchy from instToAppend until a rooted parent is found which has been assigned to a pblock (hereafter called rootedPBlock) and mark the rooted parent as zero in said array, where a rooted parent is an instance which is defined by a data object in said physical hierarchy which has a flag set which indicated said rooted parent has been assigned to a pblock and where marking an instance to zero in said array causes said instance to disappear from the physical hierarchy, where disappear from the physical hierarchy means the instance is not assigned to a particular pblock;

unwinding the recursion from said rooted parent along a line of said physical hierarchy toward said instToAppend and mark as zero in said array all ancestor instances in said physical hierarchy between said rooted parent and said instToAppend, where an ancestor instance is any instance in said physical hierarchy on a line of descendants between said rooted parent and said instToAppend not including either said rooted parent nor said instToAppend;

marking all siblings of any ancestor instance in said array as assigned to said rooted pblock if not already so marked by setting a flag in a data structure representing said instance in said physical hierarchy to a "rooted" state and

making sure a pointer to said sibling is present in a data structure representing said rooted pblock in said physical hierarchy;

determining if all sibling instances of said instToAppend in said physical hierarchy are marked as belonging to the same pblock as said instToAppend, and, if so, performing a collapse operation to resurrect a parent instance of said siblings in said physical hierarchy by removing or setting to zero entries in said array for all said sibling instances which are components of said parent instance and adding an entry to said array for said parent instance and data indicating said parent instance is assigned to the same pblock as said sibling instances which were component instances of said parent instance.

4. The process of claim 1 wherein step F is carried out by performing the following steps:

determining which instances have been moved from one pblock to another;

for each instance that has been removed from a pblock, and for each pin on the removed instance, disconnect the pin from any net to which it is connected and removing any nets not needed in a pblock, said disconnection of pins and removing of nets accomplished by altering data defining said physical hierarchy;

for each instance that has been added to a pblock, and for each pin on the added instance, create one or more new physical nets and pblock boundary pins as needed to connect said pins of all instances which have been moved to a different pblock to the same other pins said pins of said instances which have been moved were originally connected to prior to said move, said creation of new boundary pins and nets accomplished by altering data defining said physical hierarchy.

5. The process of claim 1 wherein step F is carried out by performing the following steps:

- A) determining for each pin of each instance which has been moved to a second pblock from a first pblock whether said pin is connected to a net;
- B) if so, altering data in a data object representing a pblock in which said instance to which said pin belongs was formerly assigned so as to remove said pin from a list of pins in said data object and removing said pin from a list of pins in a data object representing said net which contains a list of all pins to which said net is connected, so as to disconnect said net from said pin;
- C) determining from data in said logical netlist whether said net is connected to two or more other pins which have not been moved;
- D) if the net is not connected to two or more other pins, removing the physical net and its remaining pin by removing the remaining pin from the list of pins in a data object representing a first pblock from which said instance was removed;
- E) if said remaining pin was a boundary pin of said first pblock, locating a net in a second pblock which is connected to a boundary pin of said second pblock which boundary pin of said second pblock is connected by a net in a root pblock to said boundary pin of said first pblock and removing said net in said second pblock and removing said boundary pin of said second pblock and removing said net in said root pblock connecting said boundary pins of said first and second pblocks;
- F) if step 4C (claim 4, step C) above determines that said net is connected to two or more pins of an instance which has not been moved to another pblock, then said physical net is not removed;
- G) determine which instances have been moved from one pblock to another based upon the results of processing of step 1E (claim 1, step E) and perform a root level traversal of the original logical netlist to learn the nets that are coupled to each instance which has been moved to a new pblock;

H) create a map for all physical nets and pins that still exist in the sense that the nets have not been disconnected from pins by steps 5A through 5F above;

I) determine floating pins of instances which have been moved from one pblock to another and which have been disconnected by steps 5A through 5F above from nets to which said floating pins were formerly connected; and

J) creating one or more new recreated nets to recreate each net that was removed when an instance to which said net was connected was moved from one pblock to another so as to recreate the original connectivity defined in said logical netlist, and, if a recreated net has to cross a pblock boundary, creating one or more new nets and boundary pins on pblock boundaries with at least one recreated net connecting each pin of an instance which has been moved from one pblock to another to a boundary pin on the boundary of a pblock to which said instance is assigned and at least one new recreated net which connects a boundary pin on a boundary of a pblock to which said instance was formerly assigned to a pin to which said pin of said moved instance was formerly connected before being moved, and creating at least one new net which connects said boundary pins so as to re-establish the connection that formerly existed before said instance was moved from one pblock to another.

6. The process of claim 1 wherein step F is carried out by performing the following steps:

A) determining each pin of each instance which has been moved to a second pblock from a first pblock;

B) of the list of pins determined in step 5A, determining which pins are coupled to nets and determining all other pins to which each said net coupled to a pin determined in step 5A is also connected;

C) of the pins determined in step 5A which are connected to nets , removing said pins from a list of pins in a data object representing said first pblock;

SUBSTITUTE SPECIFICATION

D) removing the nets coupled to pins determined in step 5A by accessing a data object for each net determined in step 5B and removing the pins determined in step 5A therefrom and removing the other pins determined in step 5B therefrom;

E) determine which instances have been moved from one pblock to another based upon the results of processing of step 1E (claim 1, step E) and perform a root level traversal of the original logical netlist to learn the nets that are coupled to each instance which has been moved to a new pblock; and

F) creating one or more new recreated nets to recreate each net that was removed when an instance to which said net was connected was moved from one pblock to another so as to recreate the original connectivity defined in said logical netlist, and, if a recreated net has to cross a pblock boundary, creating one or more new nets and boundary pins on pblock boundaries with at least one recreated net connecting each pin of an instance which has been moved from one pblock to another to a boundary pin on the boundary of a pblock to which said instance is assigned and at least one new recreated net which connects a boundary pin on a boundary of a pblock to which said instance was formerly assigned to a pin to which said pin of said moved instance was formerly connected before being moved, and creating at least one new net which connects said boundary pins so as to re-establish the connection that formerly existed before said instance was moved from one pblock to another.

7. An apparatus comprising:

- a computer display;
- a keyboard and pointing device;
- a computer coupled to said display, keyboard and pointing device, and programmed to perform the following functions:
 - A) store data structures of a logical netlist;
 - B) display on one portion of said computer display a representation of instances defined by said logical netlist;

- C) provide one or more tools displayed on said computer display a user can invoke to create and locate on a floorplan of an integrated circuit being designed, said floor plan comprising one or more pblocks;
- D) respond to invocation of one or more of said tools by creating one or more said pblock(s), creation of said pblock accomplished by creating data objects representing said pblocks, said floorplan being displayed on said computer display, and wherein each pblock can contain other pblocks so as to establish a physical hierarchy;
- D) provide one or more tools a user can invoke to assign instances from said displayed representation of instances defined by said logical netlist into pblocks in said displayed hierarchy of pblocks;
- E) respond to such assignment operations by changing the data in said data objects representing said pblocks to reflect which instances are assigned to each pblock; and
- F) further respond to such assignment operations by determining the original connectivity between instances defined in said logical netlist and automatically changing data in predetermined data objects of said physical hierarchy so as to recreate said original connectivity by creating new nets and new pins as necessary which recreate said original connectivity.

8. The apparatus of claim 7 wherein said computer is programmed to allow a user to perform assignment operations of step D by allowing the user to use said pointing device to select any instance from said logical netlist and drag said instance into a displayed representation of any pblock in said physical hierarchy regardless of the logical hierarchy established by said logical netlist.

9. The apparatus of claim 7 wherein said computer is programmed to export floorplan directives regarding the physical placement on a surface of said integrated circuit being designed of all instances from said logical netlist.

10. The apparatus of claim 7 wherein said computer is programmed to carry out step E by performing the following steps:

marking instToAppend entries designating instances from said logical netlist which have been moved to a pblock with a pointer to the pblock to each instance has been moved, said pointer being in an array m_instanceAssignments (hereafter referred to as the array) which defines which instances are assigned to each pblock;

marking all child instances in said logical hierarchy of each said instToAppend in said array to point to the same pblock to which said instToAppend was assigned;

recursing up the logical hierarchy from instToAppend until a rooted parent is found which has been assigned to a pblock (hereafter called rootedPBlock) and mark the rooted parent as zero in said array, where a rooted parent is an instance which is defined by a data object in said physical hierarchy which has a flag set which indicated said rooted parent has been assigned to a pblock and where marking an instance to zero in said array causes said instance to disappear from the physical hierarchy, where disappear from the physical hierarchy means the instance is not assigned to a particular pblock; ;

unwinding the recursion from said rooted parent along a line of said physical hierarchy toward said instToAppend and mark as zero in said array all ancestor instances in said physical hierarchy between said rooted parent and said instToAppend, where an ancestor instance is any instance in said physical hierarchy on a line of descendants between said rooted parent and said instToAppend not including either said rooted parent nor said instToAppend;

marking all siblings of any ancestor instance in said array as assigned to said rooted pblock if not already so marked by setting a flag in a data structure

SUBSTITUTE SPECIFICATION

representing said instance in said physical hierarchy to a "rooted" state and making sure a pointer to said sibling is present in a data structure representing said rooted pblock in said physical hierarchy;

determining if all sibling instances of said instToAppend in said physical hierarchy are marked as belonging to the same pblock as said instToAppend, and, if so, performing a collapse operation to resurrect a parent instance of said siblings in said physical hierarchy by removing or setting to zero entries in said array for all said sibling instances which are components of said parent instance and adding an entry to said array for said parent instance and data indicating said parent instance is assigned to the same pblock as said sibling instances which were component instances of said parent instance.

11. The apparatus of claim 7 wherein said computer is programmed to carry out step F by performing the following steps:

determining which instances have been moved from one pblock to another;

for each instance that has been removed from a pblock, and for each pin on the removed instance, disconnect the pin from any net to which it is connected and removing any nets not needed in a pblock, said disconnection of pins and removing of nets accomplished by altering data defining said physical hierarchy;

for each instance that has been added to a pblock, and for each pin on the added instance, create one or more new physical nets and pblock boundary pins as needed to connect said pins of all instances which have been moved to a different pblock to the same other pins said pins of said instances which have been moved were originally connected to prior to said move, said creation of new boundary pins and nets accomplished by altering data defining said physical hierarchy.

SUBSTITUTE SPECIFICATION

12. The apparatus of claim 7 wherein said computer is programmed to carry out step F by performing the following steps:

- A) determining for each pin of each instance which has been moved to a second pblock from a first pblock whether said pin is connected to a net;
- B) if so, altering data in a data object representing a pblock in which said instance to which said pin belongs was formerly assigned so as to remove said pin from a list of pins in said data object and removing said pin from a list of pins in a data object representing said net which contains a list of all pins to which said net is connected, so as to disconnect said net from said pin;
- C) determining from data in said logical netlist whether said net is connected to two or more other pins which have not been moved;
- D) if the net is not connected to two or more other pins, removing the physical net and its remaining pin by removing the remaining pin from the list of pins in a data object representing a first pblock from which said instance was removed;
- E) if said remaining pin was a boundary pin of said first pblock, locating a net in a second pblock which is connected to a boundary pin of said second pblock which boundary pin of said second pblock is connected by a net in a root pblock to said boundary pin of said first pblock and removing said net in said second pblock and removing said boundary pin of said second pblock and removing said net in said root pblock connecting said boundary pins of said first and second pblocks;
- F) if step 4C (claim 4, step C) above determines that said net is connected to two or more pins of an instance which has not been moved to another pblock, then said physical net is not removed;
- G) determine which instances have been moved from one pblock to another based upon the results of processing of step 1E (claim 1, step E) and perform a root level traversal of the original logical netlist to learn the nets that are coupled to each instance which has been moved to a new pblock;

H) create a map for all physical nets and pins that still exist in the sense that the nets have not been disconnected from pins by steps 5A through 5F above;

I) determine floating pins of instances which have been moved from one pblock to another and which have been disconnected by steps 5A through 5F above from nets to which said floating pins were formerly connected; and

J) creating one or more new recreated nets to recreate each net that was removed when an instance to which said net was connected was moved from one pblock to another so as to recreate the original connectivity defined in said logical netlist, and, if a recreated net has to cross a pblock boundary, creating one or more new nets and boundary pins on pblock boundaries with at least one recreated net connecting each pin of an instance which has been moved from one pblock to another to a boundary pin on the boundary of a pblock to which said instance is assigned and at least one new recreated net which connects a boundary pin on a boundary of a pblock to which said instance was formerly assigned to a pin to which said pin of said moved instance was formerly connected before being moved, and creating at least one new net which connects said boundary pins so as to re-establish the connection that formerly existed before said instance was moved from one pblock to another.

13. The apparatus of claim 7 wherein said computer is programmed to carry out step F by performing the following steps:

A) determining each pin of each instance which has been moved to a second pblock from a first pblock;

B) of the list of pins determined in step 5A, determining which pins are coupled to nets and determining all other pins to which each said net coupled to a pin determined in step 5A is also connected;

C) of the pins determined in step 5A which are connected to nets , removing said pins from a list of pins in a data object representing said first pblock;

D) removing the nets coupled to pins determined in step 5A by accessing a data object for each net determined in step 5B and removing the pins determined in step 5A therefrom and removing the other pins determined in step 5B therefrom;

E) determine which instances have been moved from one pblock to another based upon the results of processing of step 1E (claim 1, step E) and perform a root level traversal of the original logical netlist to learn the nets that are coupled to each instance which has been moved to a new pblock; and

F) creating one or more new recreated nets to recreate each net that was removed when an instance to which said net was connected was moved from one pblock to another so as to recreate the original connectivity defined in said logical netlist, and, if a recreated net has to cross a pblock boundary, creating one or more new nets and boundary pins on pblock boundaries with at least one recreated net connecting each pin of an instance which has been moved from one pblock to another to a boundary pin on the boundary of a pblock to which said instance is assigned and at least one new recreated net which connects a boundary pin on a boundary of a pblock to which said instance was formerly assigned to a pin to which said pin of said moved instance was formerly connected before being moved, and creating at least one new net which connects said boundary pins so as to re-establish the connection that formerly existed before said instance was moved from one pblock to another.

14. A computer readable medium having computer executable instructions thereon for controlling a computer to perform the following steps:

- A) storing the data structures of a logical netlist;
- B) displaying on one portion of a computer display a representation of the instances defined by said logical netlist;
- C) providing one or more tools a user can invoke to create and locate on a floorplan of an integrated circuit being designed and comprising one or more pblocks and responding to the use of said tools to create one or more said

SUBSTITUTE SPECIFICATION

pblock(s) by creating data objects representing said pblocks, said floorplan being displayed on the same computer display as said representation of said instances defined by said logical netlist, and wherein pblocks can contain other pblocks so as to establish a physical hierarchy;

D) providing one or more tools a user can invoke to assign instances from said displayed representation of instances defined by said logical netlist into pblocks in said displayed hierarchy of pblocks;

E) responding to such assignment operations by changing the data in said data objects representing said pblocks to reflect which instances are assigned to each pblock; and

F) further responding to such assignment operations by determining the original connectivity between instances defined in said logical netlist and automatically changing data in predetermined data objects of said physical hierarchy so as to recreate said original connectivity by creating new nets and new pins as necessary which recreate said original connectivity.

15. The computer readable medium of claim 14 wherein said computer executable instructions include instructions to cause said computer to perform said assignment operations by allowing a user to select and drag instances from the displayed logical hierarchy into displayed pblocks and drop said instance in said pblock to assign said instance thereto by releasing a pointing tool button after the instance has been dragged to said pblock.

16. The computer readable medium of claim 14 wherein said computer executable instructions include instructions to cause said computer to also perform exporting floorplan directives regarding physical placements of all instances from said logical netlist based upon the assignment of said instances to pblocks in said physical hierarchy.

17. The computer readable medium of claim 14 wherein said computer executable instructions include instructions to cause said computer to perform step E by performing the following steps:

marking instToAppend entries designating instances from said logical netlist which have been moved to a pblock with a pointer to the pblock to each instance has been moved, said pointer being in an array

m_instanceAssignments (hereafter referred to as the array) which defines which instances are assigned to each pblock;

marking all child instances in said logical hierarchy of each said instToAppend in said array to point to the same pblock to which said instToAppend was assigned;

recurring up the logical hierarchy from instToAppend until a rooted parent is found which has been assigned to a pblock (hereafter called rootedPBlock) and mark the rooted parent as zero in said array, where a rooted parent is an instance which is defined by a data object in said physical hierarchy which has a flag set which indicated said rooted parent has been assigned to a pblock and where marking an instance to zero in said array causes said instance to disappear from the physical hierarchy, where disappear from the physical hierarchy means the instance is not assigned to a particular pblock; ;

unwinding the recursion from said rooted parent along a line of said physical hierarchy toward said instToAppend and mark as zero in said array all ancestor instances in said physical hierarchy between said rooted parent and said instToAppend, where an ancestor instance is any instance in said physical hierarchy on a line of descendants between said rooted parent and said instToAppend not including either said rooted parent nor said instToAppend;

marking all siblings of any ancestor instance in said array as assigned to said rooted pblock if not already so marked by setting a flag in a data structure representing said instance in said physical hierarchy to a "rooted" state and making sure a pointer to said sibling is present in a data structure representing said rooted pblock in said physical hierarchy;

SUBSTITUTE SPECIFICATION

determining if all sibling instances of said instToAppend in said physical hierarchy are marked as belonging to the same pblock as said instToAppend, and, if so, performing a collapse operation to resurrect a parent instance of said siblings in said physical hierarchy by removing or setting to zero entries in said array for all said sibling instances which are components of said parent instance and adding an entry to said array for said parent instance and data indicating said parent instance is assigned to the same pblock as said sibling instances which were component instances of said parent instance.

18. The computer readable medium of claim 14 wherein said computer executable instructions include instructions to cause said computer to perform step F by performing the following steps:

determining which instances have been moved from one pblock to another;

for each instance that has been removed from a pblock, and for each pin on the removed instance, disconnect the pin from any net to which it is connected and removing any nets not needed in a pblock, said disconnection of pins and removing of nets accomplished by altering data defining said physical hierarchy;

for each instance that has been added to a pblock, and for each pin on the added instance, create one or more new physical nets and pblock boundary pins as needed to connect said pins of all instances which have been moved to a different pblock to the same other pins said pins of said instances which have been moved were originally connected to prior to said move, said creation of new boundary pins and nets accomplished by altering data defining said physical hierarchy.

19. The computer readable medium of claim 14 wherein said computer executable instructions include instructions to cause said computer to perform step F by performing the following steps:

- A) determining for each pin of each instance which has been moved to a second pblock from a first pblock whether said pin is connected to a net;
- B) if so, altering data in a data object representing a pblock in which said instance to which said pin belongs was formerly assigned so as to remove said pin from a list of pins in said data object and removing said pin from a list of pins in a data object representing said net which contains a list of all pins to which said net is connected, so as to disconnect said net from said pin;
- C) determining from data in said logical netlist whether said net is connected to two or more other pins which have not been moved;
- D) if the net is not connected to two or more other pins, removing the physical net and its remaining pin by removing the remaining pin from the list of pins in a data object representing a first pblock from which said instance was removed;
- E) if said remaining pin was a boundary pin of said first pblock, locating a net in a second pblock which is connected to a boundary pin of said second pblock which boundary pin of said second pblock is connected by a net in a root pblock to said boundary pin of said first pblock and removing said net in said second pblock and removing said boundary pin of said second pblock and removing said net in said root pblock connecting said boundary pins of said first and second pblocks;
- F) if step 4C (claim 4, step C) above determines that said net is connected to two or more pins of an instance which has not been moved to another pblock, then said physical net is not removed;
- G) determine which instances have been moved from one pblock to another based upon the results of processing of step 1E (claim 1, step E) and perform a root level traversal of the original logical netlist to learn the nets that are coupled to each instance which has been moved to a new pblock;

SUBSTITUTE SPECIFICATION

H) create a map for all physical nets and pins that still exist in the sense that the nets have not been disconnected from pins by steps 5A through 5F above;

I) determine floating pins of instances which have been moved from one pblock to another and which have been disconnected by steps 5A through 5F above from nets to which said floating pins were formerly connected; and

J) creating one or more new recreated nets to recreate each net that was removed when an instance to which said net was connected was moved from one pblock to another so as to recreate the original connectivity defined in said logical netlist, and, if a recreated net has to cross a pblock boundary, creating one or more new nets and boundary pins on pblock boundaries with at least one recreated net connecting each pin of an instance which has been moved from one pblock to another to a boundary pin on the boundary of a pblock to which said instance is assigned and at least one new recreated net which connects a boundary pin on a boundary of a pblock to which said instance was formerly assigned to a pin to which said pin of said moved instance was formerly connected before being moved, and creating at least one new net which connects said boundary pins so as to re-establish the connection that formerly existed before said instance was moved from one pblock to another.

20. The computer readable medium of claim 14 wherein said computer executable instructions include instructions to cause said computer to perform step F by performing the following steps:

A) determining each pin of each instance which has been moved to a second pblock from a first pblock;

B) of the list of pins determined in step 5A, determining which pins are coupled to nets and determining all other pins to which each said net coupled to a pin determined in step 5A is also connected;

SUBSTITUTE SPECIFICATION

- C) of the pins determined in step 5A which are connected to nets , removing said pins from a list of pins in a data object representing said first pblock;
- D) removing the nets coupled to pins determined in step 5A by accessing a data object for each net determined in step 5B and removing the pins determined in step 5A therefrom and removing the other pins determined in step 5B therefrom;
- E) determine which instances have been moved from one pblock to another based upon the results of processing of step 1E (claim 1, step E) and perform a root level traversal of the original logical netlist to learn the nets that are coupled to each instance which has been moved to a new pblock; and
- F) creating one or more new recreated nets to recreate each net that was removed when an instance to which said net was connected was moved from one pblock to another so as to recreate the original connectivity defined in said logical netlist, and, if a recreated net has to cross a pblock boundary, creating one or more new nets and boundary pins on pblock boundaries with at least one recreated net connecting each pin of an instance which has been moved from one pblock to another to a boundary pin on the boundary of a pblock to which said instance is assigned and at least one new recreated net which connects a boundary pin on a boundary of a pblock to which said instance was formerly assigned to a pin to which said pin of said moved instance was formerly connected before being moved, and creating at least one new net which connects said boundary pins so as to re-establish the connection that formerly existed before said instance was moved from one pblock to another.